Research Article

A Deep Neural Network-Based Fault Detection Scheme for Aircraft IMU Sensors

Yiming Zhang¹, Hang Zhao¹, Jinyi Ma¹, Yunmei Zhao^{1,2}, Yiqun Dong¹,
and Jianliang Ai¹

¹Department of Aeronautics and Astronautics, Fudan University, Shanghai 200433, China

²School of Aerospace Engineering and Applied Mechanics, Tongji University, Shanghai 200092, China

Correspondence should be addressed to Yiqun Dong; yiqundong@fudan.edu.cn

Received 17 May 2021; Revised 26 August 2021; Accepted 1 September 2021; Published 30 September 2021

Academic Editor: Paolo Castaldi

Copyright © 2021 Yiming Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A new fault detection scheme for aircraft Inertial Measurement Unit (IMU) sensors is developed in this paper. This scheme adopts a deep neural network with a CNN-LSTM-fusion architecture (CNN: convolution neural network; LSTM: long short-term memory). The fault detection network (FDN) developed in this paper is irrelative to aircraft model or flight condition. Flight data is reformed into a 2D format for FDN input and is mapped via the net to fault cases directly. We simulate different aircrafts with various flight conditions and separate them into training and testing sets. Part of the aircrafts and flight conditions appears only in the testing set to validate robustness and scalability of the FDN. Different architectures of FDN are studied, and an optimized architecture is obtained via ablation studies. An average detecting accuracy of 94.5% on 20 different cases is achieved.

1. Introduction

Inertial Measurement Unit (IMU) is a key sensor to aircraft control system as it measures angular speeds and accelerations in flight. Faults in IMU may result in serious consequences as flight control algorithms are highly dependent on feedback of angular speeds and accelerations. Hence, it is of immense significance to achieve fault detection of IMU in order to improve fault tolerance of the control systems [1].

Hardware redundancy (HR) is a commonly adopted approach for sensor fault detection on aircrafts [2, 3]. HR scheme adopts redundant sensors and a voting system to eliminate erroneous data measured by fault sensors. Despite the fact that HR scheme usually improves fault tolerance of flight control systems, defects including high cost and vulnerability to generic fault are nonnegligible [4].

Analytical redundancy (AR) is another kind of approach, which can be divided into two categories: model-based AR and data-driven AR.

Model-based AR is a more traditional technique. Sensors are modeled mathematically, and an output estimator is pro-

posed [5]. Real output of a certain sensor is monitored and compared with the estimated output, yielding a residual, to identify sensor faults. A majority of studies on model-based analytical redundancy for fault detection were carried out by applying Kalman filtering (KF), e.g., extended Kalman filtering (EKF) [6–8], unscented Kalman filtering (UKF) [9, 10], two-step EKF [6], fuzzy logic KF [11], and hidden-Markov-based KF [12]. Other model-based methods were found using H_∞ synthesis [13–15], set-value observer [16–19], and moving horizon estimator [20]. Model-based methods are highly dependent on aircraft dynamics and kinematics, which differ significantly among different aircrafts at diverse flight conditions. Thus, scalability is not promised by using model-based schemes.

Data-driven AR is an alternative to model-based AR. As data-driven AR maps sensor outputs to fault cases directly, which means aircraft dynamics are not directly referred to in the fault detection process, it would potentially be a scalable scheme for different aircrafts and flight conditions. Neural network (NN) is widely used in recent years for data-driven AR as it is a powerful nonlinear fitting tool.

Fully connected cascade neural network is used to detect IMU failure in [21]. Adaptive EKF is compensated with online updating NN to reduce the computation time in [22]. NN-based observer is developed instead of KF for residual generating in [23, 24]. In [25], fuzzy interval models and NN are used for IMU sensor prediction. Deep neural network (DNN) is an advanced technique in fault detection as an upgraded version of NN, including recurrent neural network (RNN), long short-term memory (LSTM), and convolution neural network (CNN). Fault detection was achieved with RNN in [1, 27]. LSTM-RNN was used to denoise IMU outputs in [27]. CNN is a deep feedforward NN with representation learning capability usually used in computer vision. In [28], flight data were reformed into a 2D image-like format to use CNN for sensor fault detection. And in [29], a CNN-LSTM fusion scheme was developed for fault detection on air sensors. Other researches using NN approaches to deal with fault detection were found in [30–32].

Most existing works study fault detection problems of a certain aircraft flying a certain flight condition, e.g., cruising and land and take over (LTO). Robustness is not guaranteed. In this paper, we expect to develop a data-driven scheme to detect IMU faults, which is more robust and scalable for diverse aircrafts and flight conditions. To be specific, a CNN-LSTM-fusion fault detection network (FDN) is developed. Measured flight states reformed into a 2D image-like format is the input to the FDN and fault case classification the output. We prepared data sets of various aircrafts/flight conditions for FDN training and testing. Studies were carried out on FDN architectures starting from a “full” version that adopts all 12 flight states measured by different sensors as input. CNN and RNN are both applied for every single measured channel. Two different kinds of aircrafts at LTO and cruise stage are used for network training, and one other kind is used for testing. Robustness is justified by our diverse test sets.

2. Problem Definition

Aerodynamic is written as

$$\begin{cases} \dot{V} = (a_x - gS_\theta)C_\alpha C_\beta + (a_y + gS_\phi C_\theta)S_\beta + (a_z + gC_\phi C_\theta)S_\alpha C_\beta, \\ \dot{\alpha} = \frac{-a_x S_\alpha + a_z C_\alpha + gC_\phi C_\theta C_\alpha + gS_\theta S_\alpha}{VC_\beta} + q - \frac{(pC_\alpha + rS_\alpha)S_\beta}{C_\beta}, \\ \dot{\beta} = \frac{-(a_x - gS_\theta)C_\alpha S_\beta + (a_y + gS_\phi C_\theta)C_\beta - (a_z + gC_\phi C_\theta)S_\alpha S_\beta}{V} + pS_\alpha - rC_\alpha, \end{cases} \quad (1)$$

wherein $S_* = \sin(*)$ and $C_* = \cos(*)$. $\{V, \alpha, \beta\}$ are airspeed, angle of attack, and sideslip angle, respectively. $\{\phi, \theta, \psi\}$ and $\{p, q, r\}$ are angles and angular speeds, respectively. $\{a_x, a_y, a_z\}$ denote accelerations along the x, y, z axes in the body frame of the aircraft. External forces along body axes are defined as $\{F_k = a_k m\}_{k=x,y,z}$ which are functions of flight states, control inputs, and geometric parameters:

$$\begin{cases} F_x = F_x(\delta_{th}, V, \alpha, S, \dots), \\ F_y = F_y(\delta_r, \delta_a, V, \alpha, S, b, c, \dots), \\ F_z = F_z(\delta_e, V, \alpha, S, c, \dots), \end{cases} \quad (2)$$

wherein $\{\delta_{th}, \delta_e, \delta_a, \delta_r\}$ are control inputs of throttle, elevator, aileron, and rudder. $\{S, b, c\}$ denote wing area, wing span, and mean aerodynamic chord, respectively.

Kinematics in rotational channel are described as

$$\begin{cases} \dot{\phi} = p + \frac{qS_\phi S_\theta}{C_\theta} + \frac{rC_\phi S_\theta}{C_\theta}, \\ \dot{\theta} = qC_\phi - rS_\phi, \\ \dot{\psi} = qS_\phi + \frac{rC_\phi}{C_\theta}. \end{cases} \quad (3)$$

Dynamics in rotational channel yield [33]

$$\begin{cases} \dot{p} = c_1 r q + c_2 p q + c_3 M_x + c_4 M_z, \\ \dot{q} = c_5 p r - c_6 (p^2 - r^2) + c_7 M_y, \\ \dot{r} = c_8 p q - c_2 r q + c_4 M_x + c_9 M_z, \end{cases} \quad (4)$$

wherein $\{M_x, M_y, M_z\}$ are moments exerted on the aircraft and $c_1 \sim c_9$ are inertial parameters of the aircraft (refer to [33] for detailed definitions). External moments are functions of flight states, control inputs, and geometric parameters:

$$\begin{cases} M_x = M_x(\delta_r, \delta_a, V, \alpha, S, b, \dots), \\ M_y = M_y(\delta_r, \delta_a, V, \alpha, S, b, \dots), \\ M_z = M_z(\delta_e, V, \alpha, S, c, \dots). \end{cases} \quad (5)$$

3. Flight Data Sets

3.1. Basic Flight Data with Disturbances and Measurement Noises. To guarantee robustness and performance of the DNN-based fault detection scheme, it is vital to prepare a diverse data set. Thus, we generate a data set of 3 different aircrafts (i.e., Y, D, and B). To be specific, Y is a large cargo aircraft, D is a general aviation, and B is a passenger aircraft. We simulated 4 configurations of different flight conditions, including the following:

- (1) Landing and take-off (LTO) with manual control (aircraft Y)
- (2) High-altitude cruise with autopilot (AP) (aircraft D)
- (3) High-altitude cruise with autopilot (AP) (aircraft B)
- (4) Low-altitude free flight with manual control (aircraft B)

We intended to fully cover the whole flight envelope (e.g., LTO and cruise) by introducing the 4 configurations mentioned above. More detailed information of the configurations is concluded in Table 1.

TABLE 1: Aircrafts and flight conditions.

Aircraft	Type	Engines	Weigh	Span	Flight condition	Usage
Y	Large cargo	Quadruple piston	41.0 t	38.0 m	Low altitude, LTO, manual	Train
D	General aviation	Double piston	3.12 t	19.8 m	High altitude, cruise, AP	Train
B	Large passenger	Quadruple turbo	174 t	59.6 m	High altitude, cruise, AP	Test
B	Large passenger	Quadruple turbo	174 t	59.6 m	Low altitude, free flight, manual	Test

High-altitude cruise of aircraft Y and low-altitude LTO of aircraft D are simulated for network training; high-altitude cruise and low-altitude free flight of aircraft B are simulated for network testing.

To simulate the atmospheric turbulences, Dryden model is adopted, which injects perturbances to “clean” flight states. A speed model of Dryden wind can be defined by spectrum functions:

$$\begin{cases} \phi_u(\Omega) = \sigma_u^2 \frac{2L_u}{\pi} \frac{1}{1 + (L_u\Omega)^2}, \\ \phi_v(\Omega) = \sigma_v^2 \frac{L_v}{\pi} \frac{1 + 3(L_v\Omega)^2}{[1 + (L_v\Omega)^2]^2}, \\ \phi_w(\Omega) = \sigma_w^2 \frac{L_w}{\pi} \frac{1 + 3(L_w\Omega)^2}{[1 + (L_w\Omega)^2]^2}, \end{cases} \quad (6)$$

wherein $\{L_u, L_v, L_w\}$ represent the turbulence scale lengths. $\{\sigma_u, \sigma_v, \sigma_w\}$ represent the turbulence intensities.

Measurement noises are considered by introducing Gaussian noises. See Table 2 for noise configuration in each channel of flight states measured.

See Figure 1 for simulated flight trajectories of the 4 configurations.

3.2. Flight Data with IMU Fault Cases Injected. Typical faults of IMU include drift, noise, and scale factor. As scale factor is produced during manufacturing, which can be calibrated beforehand, we focus on noise and drift, i.e., angle random walk (ARW) and rate random walk (RRW), which behave randomly and are more difficult to detect in flight.

We introduced 4 cases with IMU faults by adding randomized faulty values to basic flight data described in Section 3.1. To be specific, 6-DoF IMU composed of triaxial accelerometer and three-axis gyroscope is studied. Although there are 3 channels in a triaxial accelerometer and three-axis gyroscope, respectively, they function as 2 measurement units. Thus, we concern about whether there are faults in the accelerometer or gyroscope, rather than in a specific acceleration or angular speed channel, to provide information for further determination if a redundant measurement unit is to take over. Fault cases are listed below. One and only one type of fault occurs simultaneously or individually in acceleration or angular speed channels.

Case 1. Noise to angular speed (i.e., $\{p, q, r\}$).

Case 2. Drift to angular speed (i.e. $\{p, q, r\}$).

TABLE 2: Measurement noise configuration.

Channel	Standard deviation	Unit
V	0.5	m/s
$\{\alpha, \beta\}$	0.0035	rad
$\{a_x, a_y, a_z\}$	0.02 g	m/s ²
$\{\phi, \theta, \psi\}$	0.002	rad/s
$\{p, q, r\}$	0.0087	rad

Case 3. Noise to acceleration (i.e. $\{a_x, a_y, a_z\}$).

Case 4. Drift to acceleration (i.e. $\{a_x, a_y, a_z\}$).

We have 5 different flight cases (including cases without IMU faults, Case 0) to be classified for each flight configuration. Fault cases happen at a random time and last for a random period (but less than 60 s) in every 60 s during flight.

Figure 2 illustrates how fault cases are injected, where red dashed lines represent basic flight data and black solid lines with existence of IMU fault. We repeat this randomized fault-injecting procedure multiple times on each flight configuration depicted in Figure 1.

3.3. Data Structure for Fault Detection Scheme. In real flights, IMU faults last for a certain length of time period with some “pattern,” as the way we insert them in Section 3.2. Therefore, it is more reasonable to observe flight data fragments rather than flight data points. A time window of length ΔT slides on each flight data fragment. Data sets composed of flight data matrices are formed. Each column of a data matrix is a vector of measured flight state at certain time t . To be specific, we downsample data fragments at sample rate of 1 Hz and use a time window of $\Delta T = 30$ s to form data matrices, yielding inputs with a dimension of 31 columns and 12 rows (representing all 12 variables measured by sensors: 3 ADS states $\{V, \alpha, \beta\}$, 3 Euler angles $\{\psi, \theta, \phi\}$, 3 angular speeds $\{p, q, r\}$, and 3 accelerations $\{a_x, a_y, a_z\}$). Figure 3 illustrates how a flight data matrix is formed in angular speed and acceleration channel, wherein the left plot is part of the flight data sequences with fault inserted, and the right plot is the flight data matrix extracted by a time window of length ΔT at time t from the sequence in the left plot.

Data sets are divided into training and testing sets. To be specific, 2/3 data of aircraft Y and D is extracted for training. 2/3 of Y and D and all data of aircraft B are for testing. Fault cases are randomly added. See Table 3 for detailed distribution of the data sets.

(a) Y, low altitude, LTO, manual

(b) D, high altitude, cruise, AP

(c) B, high altitude, cruise, AP

FIGURE 1: Continued.

(d) B, low altitude, free flight, manual

FIGURE 1: Flight trajectories of the 4 configurations in Table 1.

FIGURE 2: Example of fault cases.

FIGURE 3: Formation of flight data matrix.

TABLE 3: Data sets for DNN training and testing.

	Configuration	No. of data sets	Case distribution
Training	Y LTO manual	9160	26.9%, 18.7%, 19.9%, 17.7%, 16.7%
	D cruise AP	19420	31.4%, 17.2%, 16.3%, 17.7%, 17.4%
	Y LTO manual	4580	26.0%, 18.7%, 19.9%, 17.8%, 16.7%
Testing	D cruise AP	9710	33.5%, 20.5%, 10.5%, 19.4%, 16.1%
	B cruise AP	19420	29.7%, 16.4%, 19.4%, 16.9%, 17.5%
	B free flight manual	5328	31.5%, 16.9%, 20.1%, 19.4%, 12.1%

TABLE 4: Example of confusion matrix.

		Predicted as:				
		Clean	Angular speed drift	Acceleration drift	Angular speed noise	Acceleration noise
Actual case	Clean	95	0	1	1	2
	Angular speed drift	6	93	0	1	0
	Acceleration drift	4	0	95	0	1
	Angular speed noise	4	2	0	94	0
	Acceleration noise	4	0	2	0	94

4. IMU Fault Detection Network (FDN)

4.1. Basic Architecture of IMU FDN. The IMU fault detection problem discussed in this paper can be regarded as a classification issue of sequential data, which is composed of 2D flight-data matrices. A deep neural network with both CNN and LSTM modules is designed to deal with this kind of issue.

With CNN, we intend to extract semantic information of 2D data matrices. Feature maps are generated each time an input passes through a convolutional layer. With the increase of network layers, deeper semantic information is expected to be extracted. We use leaky ReLU as activation function which can reduce the appearance of silent neurons.

With LSTM, we aim to extract temporal information. Gate operations are used in the LSTM module to improve its performance.

4.2. Experiment Environment and Training Method. The computational platform in this work is of one i7-2600 CPU and 16GB RAM, with one Nvidia GTX 1070 GPU (driver version 456.71). The training and testing backends are based on Keras v2.4.3 and TensorFlow v2.3.0, running on Python 3.8 (Windows 10).

For network training, “crossentropy” is used to generate training loss and “Adam” is adopted as optimizer. An

exponential-decaying learning rate (lr) was adopted: $lr = 0.001 \times e^{-\text{epoch}/200}$, wherein one epoch means all training data is used once.

4.3. Evaluation of Fault Detection Performance. To evaluate the classification performance of the fault detection network, a confusion matrix is computed each time the network is tested with test sets. Each one of diagonal elements of a confusion matrix is the percentage of cases correctly predicted; i.e., the closer the diagonal elements are to 100, the better the network performs.

An example of confusion matrix is presented in Table 4. As described in Section 3.2, we took into concern both drift and noise in angular speed and acceleration channels, forming 4 fault cases and 1 clean case with no fault. Test sets (for example: aircraft Y with flight condition “low altitude, LTO, manual,” see Table 1) are sent into a fault detection network with a certain structure. Every one element of the confusion matrix is a statistical result of predictions; e.g., the 2nd row of the matrix indicates that 6% of the “angular speed drift” cases were predicted “clean,” 93 correctly predicted “angular speed drift,” 0% “acceleration drift,” 1% “angular speed noise,” and 0% “acceleration noise.” In this case, prediction precision of case “angular speed drift” is 93% for aircraft Y with flight condition “low altitude, LTO, manual.” A

TABLE 5: Prediction performance of different FDN architectures.

Architecture	Flight configuration (see Table 1)	Predicted as			
		Clean	Angular speed drift	Acceleration drift	Angular speed noise Acceleration noise
FDN-FULL		CNN + LSTM in ADS, angle, angular speed, acceleration			
Model size	Y LTO manual	92	98	87	96
122.0 MB	D cruise AP	96	99	91	94
	B cruise AP	95	97	76	95
	B free-flight manual	95	97	89	95
FDN-FULL/ADS		CNN + LSTM in angle, angular speed, acceleration			
Model size	Y LTO manual	93	95	92	96
79.7 MB	D cruise AP	96	99	91	95
	B cruise AP	95	97	83	95
	B free-flight manual	96	99	86	94
FDN-FULL/ANGLE		CNN + LSTM in ADS, angular speed, acceleration			
Model size	Y LTO manual	94	94	82	95
98.8 MB	D cruise AP	95	99	95	95
	B cruise AP	97	98	81	96
	B free-flight manual	94	96	84	95
FDN-CNN		CNN in ADS angle, angular speed, acceleration			
Model size	Y LTO manual	95	98	78	97
84.0 MB	D cruise AP	95	99	79	95
	B cruise AP	95	94	71	76
	B free-flight manual	96	96	75	94
FDN-RNN		LSTM in ADS angle, angular speed, acceleration			
Model size	Y LTO manual	67	80	65	78
46.7 MB	D cruise AP	71	84	70	76
	B cruise AP	76	79	60	64
	B free-flight manual	83	65	66	43
FDN-AB1		CNN in ADS, angle, acceleration; CNN + LSTM in angular speed			
Model size	Y LTO manual	93	98	62	97
93.7 MB	D cruise AP	96	98	91	95
	B cruise AP	93	95	72	95
	B free-flight manual	96	97	86	95
FDN-AB2		CNN in ADS, angle; CNN + LSTM in angular speed and acceleration			
Model size	Y LTO manual	97	96	88	92
60.4 MB	D cruise AP	95	99	89	92
	B cruise AP	97	97	80	94
	B free-flight manual	96	97	84	93
FDN-AB3		CNN in angle; CNN + LSTM in angular speed and acceleration			
Model size	Y LTO manual	93	98	79	99
103.0 MB	D cruise AP	96	97	83	97
	B cruise AP	95	96	68	97
	B free-flight manual	96	98	78	96
FDN-AB4		CNN in ADS, angle, angular speed; CNN + LSTM in acceleration			
Model size	Y LTO manual	96	96	80	96
93.7 MB	D cruise AP	96	99	89	95
	B cruise AP	98	98	76	96
	B free-flight manual	96	98	74	93

TABLE 5: Continued.

Architecture	Flight configuration (see Table 1)	Clean	Angular speed drift	Acceleration drift	Predicted as Angular speed noise	Acceleration noise
FDN-OPT			CNN + LSTM in angular speed, acceleration			
Model size	Y LTO manual	93	98	88	97	94
55.8 MB	D cruise AP	95	99	94	96	95
	B cruise AP	93	98	84	97	93
	B free-flight manual	96	98	92	95	95

*In this table, ADS: $\{V, \alpha, \beta\}$; angle: $\{\phi, \theta, \psi\}$; angular speed: $\{p, q, r\}$; acceleration: $\{a_x, a_y, a_z\}$.

FIGURE 4: Full architecture of the IMU fault detection network. (a) DNN-FULL architecture. ADS: airspeed, AOA, and sideslip angle measured by air data sensors; ANGLES: Euler angles; AS: angular speeds; ACC: accelerations; Conv2D: convolution layer; Dense: fully connected layer. 2×2 kernels are also used in Maxpool layers. Flatten layers reshape 2D data to 1D for Dense layers. Concatenate layer merges all the data from previous layers. (b) Detailed configurations of CNN and LSTM modules in acceleration channel, kernel size, and bias are depicted. Configurations of other channels are the same as those of the acceleration channel.

diagonal vector $[95 \ 93 \ 95 \ 94 \ 94]$ of this confusion matrix is extracted to evaluate prediction performance of the network.

With the diagonal vectors of confusion matrices discussed above, we obtain a tool to evaluate prediction performance of a certain architecture of FDN. For each network architecture to be discussed in the next section and aircrafts/flight conditions in Table 1, we send all test sets into the FDN to come to a diagonal vector of confusion matrix, generating Table 5 for comparison.

4.4. Architecture Studies of the IMU FDN. As the inputs we send into the network are not images but data sequences

of aircraft dynamics, general rules for architecture optimizing in image recognition networks may not work. We adopt ablation studies beginning with a full architecture to find a “best” network for the fault detection problem. The structure of the full network is depicted in Figure 4.

In the full network, namely, FDN-FULL, we send $\{V, \alpha, \beta\}$ (air data sensors, “ADS”), $\{\psi, \theta, \phi\}$ (Euler angles, “ANGLES”), $\{p, q, r\}$ (angular speeds, “AS”), and $\{a_x, a_y, a_z\}$ (accelerations, “ACC”) into the input layer, which means all the measured variables are used for fault detection. And for all channels, both CNN and LSTM layers are adopted.

FIGURE 5: Val. accuracy and training loss of different Nos. of CNN filters: (a) final (epoch = 2000) accuracy on validation set; (b) loss at the end of training (epoch = 2000). Each data point is an average value of 30 trainings.

FIGURE 6: Training history of different sizes of CNN kernel; (a) accuracy on the validation set; (b) loss curve during training. Each curve is an average of 30 trainings.

4.4.1. FDN-FULL/ADS. A general goal of architecture study is to maximize predict precision while minimizing net size. The FDN is essentially a nonlinear function that maps input (flight states) to output (fault cases). Notice that equation (1) indicates that accelerations (in which we are interested) are mapped to ADS data explicitly. Sending both ADS and ACC into FDN may lead to data redundancy and overfitting. So, we cut the ADS channel out so that the size of the network blocks before the Concatenate layer was reduced to 3/4 that of FDN-FULL. Prediction performance is slightly improved as expected, see Table 5.

4.4.2. FDN-FULL/ANGLES. Equation (3) indicates that Euler angle may be redundant due to the explicit mapping

relationship between it and angular speed. So, we cut ANGLE channel out just like what we did to ADS in Section 4.4.1. Prediction performance is also slightly improved while the size is reduced, as shown in Table 5.

4.4.3. FDN-CNN and FDN-RNN. We tried to cut either CNN or LSTM branches out to examine the performance of a pure CNN or RNN network. FDN-CNN performs well in some cases, but results are not satisfying when dealing with aircraft B, testing data of which is strictly separated from training set (only Y and D are used for training). That means FDN-CNN is not robust enough. FDN-RNN performs even worse, totally unable to predict faults in acceleration channel. This study led to a conclusion that

FIGURE 7: Val. accuracy and training loss of different Nos. of CNN filters: (a) final (epoch = 2000) accuracy on the validation set; (b) loss at the end of training (epoch = 2000). Each data point is an average value of 30 trainings.

FIGURE 8: FDN training history: (a) training history of different architectures (20 times of training for each architecture); (b) average loss of each architecture in 20 times of training.

both CNN and RNN are needed to construct an effective FDN for IMU fault detection.

4.4.4. FDN-OPT and Other Ablation Studies. Several other ablation studies were carried out based on FDN-FULL, trying different combinations of input data and CNN/RNN architecture, as listed in Table 5 (FDN-AB1~FDN-AB 4, FDN-OPT).

Among all the ablated FDN, FDN-OPT (with both CNN and LSTM in AS and ACC channel, ADS and ANGLES cut out) performs best and shows equivalent ability in fault

detection to FDN-FULL/ADS and FDN-FULL/ANGLE and is obviously smaller than them.

4.5. Net Hyperparameters. A major issue of constructing the FDN is to determine hyperparameters: number of CNN filters, number of LSTM nodes, and size of CNN kernel. We tried 16 different combinations of these hyperparameters based on FDN-OPT and trained 30 times with 2000 epochs for each combination to eliminate randomness in the training process. Hyperparameters are listed below:

FIGURE 9: Architecture of FDN-OPT.

Hyperparameter 1. Number of CNN filters: {8,16,32,48,64,80,96}.

Hyperparameter 2. Size of CNN kernel: $\{1 \times 1, 2 \times 2, 3 \times 3, 1 \times 1 + 2 \times 2\}$.

Hyperparameter 3. Number of LSTM nodes: {32,48,64,80,96,112,128}.

Number of CNN filters has the greatest impact on computational cost and net size. Final (at epoch 2000) validation accuracy and training loss are depicted in Figure 5. Validation accuracy increases and loss decreases with the number of filters, which means better performance. We chose 64 filters for a balance of performance and net size. Then, studies on hyperparameters 2 and 3 are carried out on a fixed number of 64 CNN filters.

The training history of different sizes of CNN kernels is depicted in Figure 6. As a common view in CNN, the size of receptive field and computational cost are proportional to the size of CNN kernels. Figure 6 shows that FDN-OPT with 2×2 or 3×3 performs better on validation sets than 1×1 or $1 \times 1 + 2 \times 2$. A model size of FDN-OPT with 2×2 CNN kernels is 55.8 MB and 3×3 59.1 MB. So 2×2 kernel is determined for balancing model size and performance.

The training history of different numbers of LSTM nodes is depicted in Figure 7. The FDN performs worst in training when the number of LSTM nodes is equal to the number of CNN kernels (64) and best when the number of LSTM nodes is half (32) or twice (128) the number of CNN kernels. As 128 nodes are a commonly used configuration in a majority of LSTM practices, it is finally adopted in our work. This result implicates that the number of LSTM nodes should not be equal to the number of CNN kernels when composing a DNN with both CNN and LSTM.

4.6. Brief Summary of FDN for IMU Fault Detection. As studied in Sections 4.4.1 and 4.4.2, discarding ADS and ANGLE channel does not make the performance of FDN worse (in some cases even better). Meanwhile, Section 4.4.3

shows that both CNN and RNN are necessary. Predicting result of FDN-OPT proved these two conclusions: for IMU fault detection problem, we need only AS and ACC information, and a CNN-LSTM fusion deep network provides satisfying performance.

Starting from FDN-FULL, we obtained an equivalent-in-performance but smaller-in-size deep network for IMU fault detection problem. We used both manual and AP flight data of 2 aircrafts (Y & D) to train the network and another aircraft (B) to test it. Results show that accuracy of prediction is over 84% for all aircrafts and flight configurations, which means the fault detection scheme is robust and potentially effective for other aircrafts/flight configurations.

To eliminate stochastic effects during NN training, we trained each architecture 20 times till loss converged (3000 epochs for FDN-RNN, 1000 epochs for other architectures). Figure 8 depicts the training history of different architectures discussed in the previous sections. In Figure 5, FDN-OPT (black lines) claims the lowest training loss on average, as shown in (b). FDN-FULL/ADS (blue lines) and FDN-FULL/ANGLE (magenta lines) perform alike, as well as FDN-FULL. Loss curves of those 4 architectures are similar in Figure 8, which strengthen the argument that prediction performances are almost of the same order as shown in Table 5. Average loss of FDN-CNN is greater than those 4 architectures but obviously much better than FDN-RNN. The reason performance of FDN-CNN being close to FDN-FULL might be that input data was arranged in a form temporal information had been included (see Section 3.3). Loss of FDN-RNN decreases much slower than any other architectures as is expected, as its prediction performance is worst in Table 5.

Both Figure 8 and Table 5 show that FDN-OPT is the most size-performance balanced one among all studied in this work; we adopt this architecture for IMU for detection task. Architecture of the optimized FDN-OPT is depicted in Figure 9.

5. Conclusion

A CNN-LSTM-fusion fault detection network (FDN) is proposed for aircraft IMU fault detection. Flight data measured by Inertial Measurement Unit (IMU) including angular speed (AS) and acceleration (ACC) are used as inputs to the FDN and fault cases as output. We simulated different aircrafts with various flight conditions to guarantee data diversity, and testing sets were data extracted from flight data that was not adopted in the training process. Testing results were satisfying for 3 different aircrafts (trained and validated with large cargo Y and general aviation D, tested with large passenger B) simulated in different flight conditions (low-altitude LTO, high-altitude AP cruise, and low-altitude manual free flight), which means the FDN developed in this paper is robust to flight conditions and potentially scalable for different types of aircrafts.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] P. Lu and E. V. Kampen, "Aircraft inertial measurement unit fault identification with application to real flight data," in *Aiaa Guidance, Navigation, & Control Conference*, 2015.
- [2] P. Goupil, "AIRBUS efforts towards advanced real-time fault diagnosis and fault tolerant control," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3471–3476, 2014.
- [3] P. Goupil, "AIRBUS state of the art and practices on FDI and FTC in flight control system," *Control Engineering Practice*, vol. 19, no. 6, pp. 524–539, 2011.
- [4] R. J. Patton, *Fault-tolerant control systems: the 1997 situation*, The University of Hull, 1997.
- [5] I. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, "Model-based fault diagnosis for aerospace systems: a survey," *Proceedings of the Institution of Mechanical Engineers Part G Journal of Aerospace Engineering*, vol. 226, no. 10, pp. 1329–1360, 2012.
- [6] Q. He, W. Zhang, X. Liu, and J. Liu, "Information fusion and reconstruction of key sensors in a flight control system in constant wind field based on two stage EKF," in *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pp. 718–724, 2017.
- [7] L. V. Eykeren and Q. P. Chu, "Sensor fault detection and isolation for aircraft control systems by kinematic relations," *Control Engineering Practice*, vol. 31, pp. 200–210, 2014.
- [8] M. Ariola, F. Corrado, M. Mattei, I. Notaro, and A. Sollazzo, "An SFDI observer-based scheme for a general aviation aircraft," in *Control and Fault-Tolerant Systems (SysTol), 2013 Conference on*, pp. 149–158, 2013.
- [9] P. Lu, L. V. Eykeren, E. Kampen, Q. P. Chu, and B. Yu, "Adaptive hybrid unscented Kalman filter for aircraft sensor fault detection, isolation and reconstruction," in *Aiaa Guidance, Navigation, & Control Conference*, 2014.
- [10] M. Ling and Y. Zhang, "Fault detection and diagnosis for GTM UAV with dual unscented Kalman filter," in *Aiaa Guidance, Navigation, & Control Conference*, 2015.
- [11] D. Berdjag, A. Zolghadri, and J. Cieslak, "Fault detection and isolation of aircraft air data/inertial system," in *European Conference for Aerospace Sciences*, pp. 317–332, 2013.
- [12] K. Rudin, G. Ducard, and R. Y. Siegwart, "Sensor Fault Detection for Aircraft Using a Single Kalman Filter and Hidden Markov Models," in *IEEE Conference on Control Applications, CCA, 2016*, pp. 991–996, IEEE, 2014.
- [13] P. Freeman, P. Seiler, and G. J. Balas, "Air data system fault modeling and detection," *Control Engineering Practice*, vol. 21, no. 10, pp. 1290–1301, 2013.
- [14] M. Mattei and G. Paviglianiti, "Managing sensor hardware redundancy on a small commercial aircraft with H_∞ FDI observers," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 347–352, 2005.
- [15] F. Amato, C. Cosentino, M. Mattei, and G. Paviglianiti, "A direct/functional redundancy scheme for fault detection and isolation on an aircraft," *Aerospace Science and Technology*, vol. 10, no. 4, pp. 338–345, 2006.
- [16] Y. Wan and T. Keviczky, "Implementation of real-time moving horizon estimation for robust air data sensor fault diagnosis in the RECONFIGURE benchmark," *Ifac Papersonline*, vol. 49, no. 17, pp. 64–69, 2016.
- [17] P. Rosa, C. Silvestre, J. S. Shamma, and M. Athans, "Fault detection and isolation of LTV systems using set-valued observers," *IEEE*, 2010.
- [18] P. Casau, P. Rosa, S. M. Tabatabaeipour, and C. Silvestre, "Fault detection and isolation and fault tolerant control of wind turbines using set-valued observers," *Ifac Proceedings Volumes*, 2012.
- [19] P. Rosa and C. Silvestre, "Fault detection and isolation of LPV systems using set-valued observers: an application to a fixed-wing aircraft," *Control Engineering Practice*, vol. 21, no. 3, pp. 242–252, 2013.
- [20] Y. Wan and T. Keviczky, "Real-time fault-tolerant moving horizon air data estimation for the RECONFIGURE benchmark," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 3, pp. 997–1011, 2019.
- [21] S. Hussain, M. Mokhtar, and J. M. Howe, "Sensor failure detection, identification, and accommodation using fully connected cascade neural network," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1683–1692, 2015.
- [22] A. Abbaspour, P. Aboutaleb, K. K. Yen, and A. Sargolzaei, "Neural adaptive observer-based sensor and actuator fault detection in nonlinear systems: application in UAV," *ISA Transactions*, vol. 67, pp. 317–329, 2017.
- [23] U. Papa, M. L. Fravolini, G. D. Core, P. Valigi, and M. R. Napolitano, "Data-driven schemes for robust fault detection of air data system sensors," *IEEE Transactions on Control Systems Technology*, vol. 27, 2019.
- [24] S. Gururajan, M. L. Fravolini, H. Chao, M. Rhudy, and M. R. Napolitano, "Performance evaluation of NN based approaches for airspeed sensor failure accommodation on a small UAV," in *Mediterranean Conference on Control and Automation*, pp. 603–608, 2013.
- [25] M. Crispoltoni, M. Fravolini, F. Balzano, S. D'Urso, and M. Napolitano, "Interval fuzzy model for robust aircraft IMU sensors fault detection," *Sensors*, vol. 18, no. 8, p. 2488, 2018.

- [26] E. Sobhani-Tehrani, H. A. Talebi, and K. Khorasani, "Hybrid fault diagnosis of nonlinear systems using neural parameter estimators," *Neural Networks*, vol. 50, pp. 12–32, 2014.
- [27] C. Jiang, S. Chen, Y. Chen et al., "A MEMS IMU de-noising method using long short term memory recurrent neural networks (LSTM-RNN)," *Sensors*, vol. 18, no. 10, p. 3470, 2018.
- [28] Y. Dong, "An application of deep neural networks to the in-flight parameter identification for detection and characterization of aircraft icing," *Aerospace Science & Technology*, vol. 77, pp. 34–49, 2018.
- [29] Y. Dong, "Implementing deep learning for comprehensive aircraft icing and actuator/sensor fault detection/identification," *Engineering Applications of Artificial Intelligence*, vol. 83, pp. 28–44, 2019.
- [30] Y. C. Choi, J. H. Son, and H. S. Ahn, "Fault detection and isolation for a small CMG-based satellite: a fuzzy Q-learning approach," *Aerospace Science and Technology*, vol. 47, pp. 340–355, 2015.
- [31] A. Shakouri and N. Assadian, "Fault detection and isolation of satellite gyroscopes using relative positions in formation flying," *Aerospace Science & Technology*, vol. 78, pp. 403–417, 2018.
- [32] B. Eroglu, M. C. Sahin, and N. K. Ure, "Autolanding control system design with deep learning based fault estimation," *Aerospace Science and Technology*, vol. 102, p. 105855, 2020.
- [33] B. L. Stevens, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*, New York: Wiley, 2015.